Decontamination from Black Virus Using Parallel Strategy

by

Yichao Lin

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In this thesis, the problem of decontaminating networks from *black virus* (BVs) using parallel strategy with a team of system mobile agents (the BVD problem) is studied. The BV is a harmful process whose initial location is unknown a priori. It destroys any agent arriving at the network site where it resides, and once triggered, it spreads to all the neighboring sites, i.e, its clones, thus increasing its presence in the network. In order to permanently remove any presence of the BV with as less execution time as possible and minimum number of site infections (and thus casualties), we propose parallel strategy to decontaminate the BVs: instead of exploring the network step by step we employ a group of agents who follow the same protocol to explore the network at the same time, thus dramatically reducing the time needed in the exploration phase and minimizing the casualties. Different protocols are proposed in meshes, tori, and chordal rings following the monotonicity principle. Then we analyze the cost of all our solutions and compare to the asynchronous BV decontamination. Finally conclusion marks are presented and future researches are proposed.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A distributed system is a group of computational entities cooperating with each other to achieve one or more tasks. This thesis deals with distributed computing by mobile agents in network. More specifically, we deal with the problem of deploying a group of mobile agents who follow the same protocol to explore the network and decontaminate the dangerous virus (called Black Virus) present on the network nodes. In this chapter, the motivations of the problem are provided, following is a brief summary of the contributions.Finally, an overview of the organization of the thesis is presented.

1.1 Problem and Motivation

Mobile agents are widely used in distributed and network systems while the applications of them can cause some security issues, thus threatening to the network: A contaminated or infected host can destroy working agents for various malicious purposes; A malicious agent can contaminate or infect other computer nodes so they become malfunctional or crash. The harmful hosts, often called *Black Holes* trigger the problem called *Black Hole Search* (BHS), the focus of which is to locate their positions since it is statics. This problem has been studied in many variants. For example, different topologies and different settings

(synchronous and asynchronous). The harmful agents trigger the problem called *Intruder Capture* (IC). Its main focus is to deploy a group of mobile agents to capture a extraneous mobile agent (the intruder) who moves arbitrarily fast through the network and infects the visiting sites. Also it has been investigated in a variety of topologies. More detailed literature review will be provided in Chapter 2. Note that BH is static and only damage the agents reaching it without leaving any detectable trace. Intruder is mobile and harmful to the network nodes but does not cost any harm to other system agents. A new harmful presence called *black virus* BV has been initially introduced by Cai et al. in[**?**]. It is a dangerous process resides at an unknown site in a network and destroys any upcoming agents, but unlike the BH, the node where the original BV resides thus become clean. At the same time, the original BV multiplies (called clones) and spread to all neighbouring nodes, thus increasing its number,and damage in the network. A BV is destroyed when it moves to a site where there is already an agent. Based on this harmful presence, a new problem called Black Virus Decontamination(BVD) is presented by Cai et al., the main focus of which is to use a group of system agents to permanently remove any presence of the BV from the network. A protocol defining the actions of the agents solves the BVD problem if at least one agent survives and the network is free of BVs. Also, a desirable property of a decontamination protocol is that the nodes which have been explored or cleaned by mobile agents are not be recontaminated by the BV spreading. A solution protocol with such a property will be called *monotone*. see[**?**]. Some important cost measure is the number of node infections by the BVs (casualties); size of the team, i.e, the number of agents employed by the solution, the time needed by the solution. Solutions in which the agents explore the network's node in sequence have been proposed in [**?**], [**?**] and [**?**]. The size of the team is minimum in [**?** **?**] and the number of site infections is also minimum in such case, i.e., exploring the network nodes in sequence but the units of time that these protocol (including the protocol in chordal rings) cost is usually several times as much as the total number of the network's nodes (assuming in synchronous setting). Now

we are interested in the solution using parallel strategy where we deploy a larger number of mobile agents following the same protocol to decontaminate the network in the exploring phase with the goal to minimize the *total working time* (TWT) which is calculated by multiplying the number of agents and the total execution time and also the casualties.

1.2 Our Contribution

1. In this thesis, we propose parallel strategy to solve the BVD problem. It is the first attempt to deal with this issue in a parallel way. Agents are not allowed to communicate with each other unless they are in the same network node so the protocol should enable the agents in different nodes to move properly, i.e, the route of every agent is different but they are served to explore the network; when a BV is triggered, other agents should bypass the new-formed BVs... We give simple but efficient solution to deal with this problem with acceptable cost. Also we give the size of the exploring team which is minimum to guarantee both the TWT and the casualties we reach.

2. The BVD problem is investigated for three important topologies: *meshes*, *tori*, *chordal rings*. All the protocol are optimal both in term of TWT and casualties. We compare our solution with [**?**] and [**?**] in which the exploring route is in sequence and the result is that our solution is better than that of them in terms of TWT and casualties. One should be point out that in chordal ring especially, the more complicated the chordal ring becomes, the more TWT that we save comparing to [**?**].

1.3 Thesis Organization

The thesis is organized as followed:

Chapter 2 contains a literature review on related problems. We begin by reviewing the Black Hole Search and Intruder Capture problem, and then focus on the solution of BVD problem where the mobile agents explore the network in sequence, the issue has been studies in different topologies: two-dimensional grids, three-dimensional grids, tori, chordal rings, hypercubes and arbitrary network. Also, the variant of this problem, which is decontamination of an arbitrary network from multiple black virus is also reviewed.

Chapter 3 introduces terminology, definitions and model for the BVD problem used in the rest of the thesis. Also we describe the high level ideas that serve as the basic of all our solutions. Since monontone is the necessary condition for spread optimality, we go through the principle and finally make a conclusion.

Chapter 4 focus on the BVD problem for the mesh topology. In this chapter, an optimal algorithm in terms of casualties and TWT is developed. Complexity analysis in terms of casualty and TWT are performed and obtained. Some comparison and analysis are also made between our solution and [**?**] and the result shows that our solution is better.

Chapter 5 presents the BVD problem for the chordal ring topology. In this chapter, we introduce the *Three Jump Notifying Technique* (TJNT) to manipulate each mobile agent efficiently go through their route in the exploring phase and avoid any new-formed BV after the original BV is triggered. Based on this technique, we develop the parallel strategy for the mobile agents to decontaminate the choral ring from BV. Complexity analysis in terms of casualty and TWT are performed. Finally some comparison and analysis are made between our solution and [**?**] and the result shows that our solution performs better.

Chapter 6 summaries the main conclusion of our work and present some open problems and future work.

Chapter 2

Literature Review

Mobile agents have been widely used in the field of distributed computing due to their features especially the mobility which allow them to migrate between computers at any time during their execution. A group of agents can be used to perform a various tasks, for example, network exploration, maintenance, and etc. However, the introduction of mobile agent tend to cause security problem, thus threatening the network. Various security issues and solution algorithms have been proposed by Flocchini and Santoro in [**?**]. Generally, the threaten that the mobile agents cause are divided into two categories: in first case,the malicious agents can cause network nodes malfunction or crash by contaminating or infecting them (the harmful agent); in second case, the contaminated or infected hosts can destroy working agent for various malicious purposes (the harmful hosts). These two threaten trigger two problems: Black Hole Search (BHS) and Intruder Capture (IC) which will be introduced in the following sections. Then we review the BVD problem which deal with the decontamination of a harmful presence which cause the network node malfunction but leaves the network node clean when it is triggered and spreads to all its neighbouring nodes, thus increases it presences. In the section introducing BVD problem, we present the the abilities of mobile agents that has been proposed and different decontamination strategies based on different strategies.

2.1 Black Hole Search, BHS

The BHS problem assumes there is a BH or multiple static BHs residing at certain network nodes and will destroy any upcoming agents without leaving any detectable trace.The task is to use a team of agent to locate the black hole(s) and is completed when at least one agent survives and reports the location(s) of the black hole(s). Note that the solution is based on graph exploration and the goal can be reached totally depending on the sacrifice of some agents. In [**?**], Das et al. considered a model for unknown environment with dispersed agents under the weakest possible setting, many exploration models and works were included in this article. The BHS problem has been widely studies in various topologies and settings: the timing is synchronous or asynchronous; the number of black hole(s) is known or unknown. For example: by Chalopin [**?** **?**] in asynchronous rings and tori, Dobrev et al. [**?**] in arbitrary graph, [**?**] in anonymous ring and [**?**] in common interconnection networks...What is worth pointing out is that the number of BHs remains the same as it of the beginning, thus not causing harm to other sites of the network.

2.2 Intruder Capture

The IC problem assumes that there is an intruder moves with an arbitrary speed from node to node in the network and contaminate the sites it visits, the goal of which is to deploy a group of mobile agents to capture the intruder;the intruder is captured when it comes in contact with an agent. Note that the intruder does not cause any harm to the upcoming agents. It is equivalent to the problem of decontaminating a network contaminated by a virus while avoiding any recontamination. This problem is first introduced in [**?**] and has been widely investigated in a variety network topologies: trees [**?** **?** **?**], hypercubes[**?**], multi-dimensional grids[**?**], chordal rings[**?**] etc. The studies of arbitrary graph has been started in [**?** **?**]. Note that monotone is a critical principle in the solutions of IC

problems.

2.3 Agent Capabilities

Different capacities granted to the mobile agents have an impart on solving the BHS problem,IC problem and also the BVD problem.now we discuss these capabilities in the following section.

Communication Mechanisms Mobile agents can communicate with each other only when they are in the same node in a network.Some essential communication methods have been studied in literature: whiteboard,tokens and time-out. In [**?** **?** **?** **?**], the whiteboard model is used, which is a storage space located at each node and agents arriving there are able to read and write. In the token model, (see [**?** **?**]), tokens are like memos of the agents which can be dropped off and picked by agents at nodes or edges. While the time-out mechanism can only be used in synchronous setting where each agent has a pre-determined amount of time. (see [**?** **?** **?**]).

Knowledge of the topology Different assumption of mobile agents' knowledge of the topology has an impact on solutions of some of the problems mentioned above, for example, the BHS problem. In [**?**], Dobrev et al. present three types of topological knowledge in an asynchronous arbitrary network and show the results of the BHS problem based on different setting of the topology knowledge.

Other capabilities In some studies, agents are endowed with the visibility, which mean that they can see whether or not their neighbouring nodes are clean or contaminated (see [**?** **?**]).They observe that the visibility assumption allow them to drastically decrease the time and move complexities in torus, chordal ring and hypercubes when dealing with IC

problem. For example, in chordal ring $C_n\{d_1 = 1, d_2, ..., d_k\}$, the number of agents, the time and the moves required in local model are $(2d_k + 1)$, $3n - 4d_k - 1$, $4n - 6d_k - 1$ respectively, while in visibility model, they are $2d_k$, $\left\lceil \frac{n-2d_k}{2(d_k - d_{k-1})} \right\rceil$, $n - 2d_k$. In torus , the number of agent, the time and the moves required are $2h + 1$, $hk - 2h$, $2hk - 4h - 1$ and in visibility model are $2h$, $\lceil \frac{k-2}{2} \rceil$, $hk - 2h$ respectively. They also compare the complexity of both models in hypercubes, a algorithm requiring Θ $(\frac{n}{\sqrt{logn}})$ number of agents and $O(n \log n)$ moves while the algorithm they propose in the visibility model requires $\frac{n}{2}$ agents and $O(n \log n)$ moves.

In [**?**], the concept of *k-hop visibility* is presented. The agents have the *full topologies* if each of them have a map in their memory of the entire network including the identities of the node and the labels of the edges. If a agent has *k-hop visibility*, then at a node v a agent can see the k-neighbourhood $N^{(v)}$ of v, including the node identities and the edge labels. Note that *Diam-hop* visibility is equivalent to full topological knowledge.

Another interesting capability of agents is cloning which is introduced in [**?**]. Cloning is the capacity for an agent to create copies of itself. In this paper, they also discuss how the combination of different capacities reaches different optimal strategy in IC problem in hypercube. For example, the strategy is both time and move-optimal when visibility and cloning are assumed or when cloning, local and synchronicity are assumed. But the time and move-optimal strategy can be obtained at the expense of increasing the number of agents. The last capability of agents be discussed is immunity which means that a node is immune from recontamination after an agent departs. Two kinds of immunity have been proposed:local and temporal. In local immunity, (see [**? ?**], the immunity of a node depends on the state of its neighbouring nodes. More specifically, a node remains clean after the departure of an agent until more than half of its neighbours are contaminated. In the temporal immunity, a node is immune for a specific amount of time (t). The node remains clean until time expires and becomes recontaminated if at least one of its neighbours are contaminated. In models without immunity assumption, a node becomes recontaminated if it has at least one contaminated neighbours.

2.4 Black Virus Decontamination

2.4.1 Overview

The BVD problem is first introduced by Cai at al. in citeCai: A black virus is a extraneous harmful process endowed with capabilities for destruction and spreading. The location of the initial BV(s) is known a priori. Like a BH, a BV destroy any agents arriving at the network where it resides. When that happen, the clones of the original BV spread to all it neighbouring nodes and remain inactive until an agent arrives. The BVD problem is to permanently remove any BVs in the network using a team of mobile agents. They proposed that the only way to decontaminate a BV is to surround all its neighbouring nodes and send an agent to the BV node. In this case, the node where the original BV resides is clean and all its clones are destroyed by the guarding agents in its neighbouring nodes.They have presented different protocols in various topologies: q-grid, q-torus, hypercudes in [**?**] and arbitrary graph [**?**]. A basic idea of implementing the decontamination has also been propose by them assuming that the timing is asynchronous which divides the whole decontamination process into two part: "shadowed exploration"and "surround and eliminate". In order to minimize the spread of the virus, they use a "safe-exploration" technique which is executed by at least two agents: the "Explorer Agent" and the "Leader Explorer Agent" who both residing at a safe node u at the beginning, for example,the homebase. The Explorer Agent moves to a node v to explore it and it needs to return to node u to report the node v is safe. The "Leader Explorer Agent" determines if the node v is safe or not by "Explorer Agent "s arriving or a BV's arriving. If node v is safe, both of them move to node v. For the purpose of insuring monotonicity, at any point in time the already explored nodes must be protected so they are not be recontaminated again. After the BV is detected, the "surround and eliminate" begins. In this phase, some agents are employed to surround the new-formed BVs (the clones of the original BV) then some agents are sent to the clones to permanently destroy them. This is called the "Four-step

Cautious Walk" and is widely used in BVD problem with synchronous setting. Also, BVD problem in chordal ring has been discussed in [**?**].

2.4.2 BVD in different topologies

Protocols regarding to BVD problems in grid are BVD-2G and BVD-qG which deal with BVD problems in 2-dimensional grid (meshes) and q-dimensional grid respectively. BVD-2G performs a BV decontamination of a 2-dimensional grid of size n using $k = 7$ agents and 3 casualties, within at most $9n+O(1)$ moves and at most $3n$ time. While protocol BVD-qG performs a decontamination of a q-dimensional grid of size $d_1 \times d_2 ... \times d_q$ using $3q+1$agents and at most $q+1$ casualties, within at most $O(qn)$ moves and at most $\Theta(n)$ time. Algorithm to decontaminate the BV in a q-dimensional torus,called BVD-qT uses $4q$ agents with 2 casualties with at most $O(qn)$ moves and $\Theta(n)$. Protocol BVD-qH is to perform a BV decontamination of a q-hypercube using $2q$ agents and q casualties with at most $O(n \log n)$ moves and $\Theta(n)$. In arbitrary graph G(see [**?**]), two protocols are presented: GREEDY EXPLORATION and THRESHOLD EXPLORATION. In these two protocols, $\Delta+1$ agents are needed and both of the protocols are worst-case optimal with respect to the team size where Δ represents the maximum degree in G. Though the protocols are described for a synchronous setting, they easily adapt to asynchronous ones with an additional $O(n)$ moves for the coordinating activities. An advantage of these protocols is that the agents can use only local information to execute the protocol. Another interesting fact based on these two protocols is that both GREEDY ROOTED ORIENTATION and THRESHOLD ROOTED ORIENTATION produce an optimal acyclic orientation rooted in the homebase.

In [**?**], solution for BVD in chordal ring is discussed. In Alotaibi's thesis, she discuss solutions based on different kinds of chordal ring: double loops, triple loops, consecutive-chords rings and finally general chordal ring. In double loops, she proposed three strategies in elimination phase and the upper bound of moves is $4n - 7$ in the whole protocol and

a maximum of 12 agents are employed.In triple loops, she discusses two classes of chordal ring: $C_n(1, p, k)$ and $C_n(1, k-1, k)$. In any triple loop $C_n(1, p, k)$, a maximum of $5n - 6k + 22$ moves and 24 agents are needed for the decontamination while in any triple loop $C_n(1, k-1, k)$, a maximum of $5n - 7k + 22$ moves and 19 agents are needed. Finally in the consecutive-chords ring, a maximum of $(k+2)n - 2k - 3$ moves and $4k + 1$ agents are needed. She described the decontamination strategies in synchronous setting but only with a cost of $O(n)$ moves can the strategies be used in asynchronous setting.

Chapter 3

Definitions and Terminology

3.1 Model

3.1.1 Network, Agent, Black Virus

Network The environment in which mobile agents operate is a network modelled as simple undirected connected graph with $n = |V|$ nodes (or sites) and $m = |E|$ edges (or links). We denote by $E(v) \subseteq E$ the set of edges incident on $v \in V$, by $d(v) = |E(v)|$ its degree, and by$\Delta(G)$ (or simply Δ) the maximum degree of G. Each node v in the graph has a distinct $id(v)$. The links incident to a node are labelled with distinct port numbers. The labelling mechanism could be totally arbitrary among different nodes; without loss of generality, we assume the link labelling for node v is represented by set $l_v = 1, 2, 3, ..., d(v)$.

Agent A group of mobile agents are employed to decontaminate the network. The agent is modelled as a computational entity moving from a node to neighbouring node. More than one agents can be at the same node at the same time. Communication among agents occurs at this time; there are no a priori restrictions on the amount of exchanged information. In our thesis, we employ two groups of agents (the exploring group and the shadowing group) operating the same protocol and we assume that all the agent?s moves follow the same

clock. Agents in the exploring group Also, agents are endowed with 1-hop visibility which means at a node v, it can see the labels of the edges incident to it and the identities of all its 1-hop neighbours. We do not guarantee other capability introduced in Chapter2 to the agents in our protocols.

Black Virus In G there is a node infected by a black virus (BV) which is a harmful process endowed with reactive capabilities for destruction and spreading. The location of the BV is not known at the beginning. It is not only harmful to the node where it resides but also to any agent arriving at that node. In fact, a BV destroy any agent arriving at the network site where it resides, just like the black hole. Instead of remaining static as the black hole, the BV will spread to all the neighbouring sites leaving the current node clean. The clones can have the same harmful capabilities of the original BVs (fertile) or unable to produce further clones(sterile). A BV will be destroyed if and only if the BV arrive at a node where there is already an agent. Thus, the only way to eliminate the BV is to surround it completely and let an agent attack it. In such situation, the attacking agent will be destroyed while the clones of the original BV will be permanently eliminated by the agents residing the neighbouring nodes of the original node. We assume that at the same node, multiple BVs (clone or original) are merged. More precisely, at any time, there is at most one BV at each node. Another important assumption is that when a BV and an agent arrive at an empty at the same time, the BV dies and the agent survive remaining unharmed.

Summarizing, there are five possible situations when an agent arrive at a node v:

- agents arrive at a node which is empty or contains other agents, they can communicate with each other and the node v is clean.
- agents arrive at a node which contains a BV, the agent dies and the clones of the BV (BVC) spread to all the neighbours of v leaving node v clean.

- A BVC arrives at a node which is empty or there is already a BV: the node becomes/stays contaminated; it merges with other BVs.
- BVCs arrive at a node v which contains one or more agents, the BVCs are destroyed but the agents are unharmed.
- A BVC and an agent arrive at an empty node at the same time, the BVC dies while the agent remains unharmed.

3.1.2 Problem, Cost

The BLACK VIRUS DECONTAMINATION (BVD) problem is to permanently remove the BV, and its clones from the network using a team of mobile agents starting from a given node, called home base(HB). The solutions where the agents explore the network sequentially have been proposed in some classes of topologies. chordal rings, hypercubes and arbitrary graph. In this thesis, we are interested in parallel strategies in BVD problem: instead of exploring the network in sequence, we explore it in parallel; in chordal ring, we also propose a parallel solution to surround the clones of the original BV. In this thesis, the efficiency measurements we have are: *spread* of BV (also measures the number of agents *casualties*; the *size* of the team, i.e, the number of agents employed by the solution; total working time(TWT) (calculated by multiplying the *size* of the team and the time cost by the solution. Note that TWT does not contains any practical meaning but exist only as a measurement. We propose TWT to compare more fairly the time of two protocols when the number of agents is different.

3.1.3 Monotone, Synchrony

A desirable property of a decontamination protocol is to prevent the nodes which been explored or cleaned by mobile agent from being recontaminated which will occur if the

clones of the BV are able to move to a explored node in absence of agent. A BVD protocol with such property is called monotone. Monotone property is the necessary condition for spread optimality.

Asynchrony refers to the execution timing of agent movement and computations. The timing can be *Synchronous* or*Asynchronous*. When the timing is synchronous, there is a global clock indicating discrete time unit; it takes one unit for each movement (by agent or BV); computing and processing is negligible. When we have asynchronous agents, there is no global clock, and the duration of any activity (e.g., processing, communication, moving) by the agents, the BV, and its clones is finite but unpredictable. In this thesis, all our protocols work in synchronous setting.

3.2 General strategy

Following the solution in sequential case, we decomposed the BVD process into two separate phase: *Shadowed Exploration* and *Surrounding and Elimination*. The task of the first phase is to locate the BV and the second phase is to decontaminate the BV and its clones. Apart from these two basic phase, we have initialization which is to deploy the agents properly at the beginning of executing the protocol because we explore the network in parallel, and the arrangement of the agents is crucial to successfully execute the protocol.

Phase1: Shadowed Exploring Agents employed are divided into two group: shadowing group and exploring group, and the number of agents in two group is the same. For convenience, we call the agent in the shadowing group the shadow agent (SA and those in the exploring group the exploring agent EA (one EA is accompanied by one SA. As the name indicated, agents in exploring group explore the network and the agents in shadowing group follow the agents protecting the node which have been explored. More precisely, at T_1, EA moves to node v; at T_2, SA moves to node v and EA moves to node u supposing that node v is clean.

Phase2: Surrounding and Elimination In this phase, since we already know the position of the BV, we employ agents to surround all the neighbours of the BVCs. Once all the agent arrive the proper positions(all the neighbours of the BVCs are guarded), we employ another group of agents (the number of them is equal to the number of BVCs) to move to the BVCs, thus permanently destroy them. Usually, some of the agents moving to the neighbours of the BVCs are from the shadowing group and exploring group because in this way we can save the number of agents used in the whole protocol. Note that not all the agents are informed when the BV is detected. More specifically, only the agents who receives the clones know the existence of the BV, and other agents keep moving in the network. In some simple topologies, such as meshes and torus, the second phase begins when the BV is detected since the number of agents are enough to proceed the second phase. In some more complicated topologies, for example, the chordal ring which we discuss later, we take some other measures to call back enough number of agents to finish the second phase.

3.3 Evaluation Criteria

For each of our strategy, we compute the number of agents, the number of movements, the time and the casualties. In addition, we propose another cost measure which combines all the criteria mentioned above (Cost Function Measurement). In this thesis, we propose parallel strategy which aim at reducing the execution time and casualties. We reach this purpose at the cost of employing more agents to explore the graph parallelly. Unlike the sequential strategy which use only one agent, the parallel strategy are more flexible because you can adjust the strategy based on your resource (the number of agents, the time,...). Generally, the more agents we employ to explore the graph, the less time we use in the shadowed exploration phase(the number of casualties may reduce at the same time), so there are some tradeoff we should make when we want to use the parallel BV decontamination

strategy in practice. Now we introduce the general idea of the Cost Function Measurement. Let's denote by w the number of agents we employ in the strategy, by x the number of movements, by y the time and by z the casualties. Additionally, we introduce three useful function: $Cost_{AgentNumber}(w)$ which given the number of agents employed in the strategy, returns the overall cost of these agents; $Cost_{Movements}(x)$ which given the number of movements needed to complete the strategy, returns the overall cost of the movements; $Cost_{time}(y)$ which given the time required to finish the protocol, returns the overall cost of the time; $Cost_{casualty}(z)$ which given the casualty of the strategy, returns the overall of the casualties. Then the Cost Function $C(w, x, y, z)$ can be presented as $C(w, x, y, z) = C(Cost_{AgentNumber}(w), Cost_{Movements}(x), Cost_{time}(y), Cost_{casualty}(z))$. Users can build their own function to fit the specific scenario. For example, if the user care more about the time cost, then they can increase the weight of the time; they can also adjust each subfunction to simulate the practical situation. The simplest model is the liner function. More specifically, the cost functions can be expressed as below:

$$\begin{cases} C(w, x, y, z) = Cost_{AgentNumber}(w) + Cost_{Movements}(x) + Cost_{time}(y) + Cost_{casualty}(z) \\ Cost_{AgentNumber}(w) = \alpha w + \alpha_1 \\ Cost_{Movements}(x) = \beta x + \beta_1 \\ Cost_{time}(y) = \gamma y + \gamma_1 \\ Cost_{casualty}(z) = \delta z + \delta_1 \\ w, x, y, z \in \mathbb{Z} \end{cases} \tag{3.1}$$

In this function we can see that the user view the four factor evenly (the coefficients of the four subfunctions are the same which are "1"), and the cost of each item(agents, movement, time, casualty) increase linearly with their independent variable. We use this liner model to analyze some of our strategies. Note that the function we present here are for the purpose of explaining how to use the function method to analyze the BVD strategies,

the function itself can be adjust by the users based on the practical situation. We simply compare the overall cost of different strategies for the same problem and assume that the strategy with the minimum cost is the best solution for the users.

3.4 Conclusion

In this Chapter, we presented the model of our problem and also some important terminologies. Also we described a general strategy for our problem depending on particular setting: synchronous timing, parallel strategy... In the next chapter, we discuss the parallel strategies in BVD problem in two simple topologies: meshes and torus.

Chapter 4

Parallel Black Virus Decontamination in Meshes

4.1 Introduction

In this chapter, we discuss parallel strategy on BVD problem in grid and tori. In the sequential strategy (*BVD-2G*) for 2-dimensional grids which are meshes, an "explorer agent "and a "leader explorer agent" are sent to explore the graph and locate the BV. They traverse the mesh in a snake-like fashion column by column following the "casual walk". In the sequential protocol (*BVD-qG*) for q-dimensional grids, the grid is partitioned into $d_1 \times \ldots \times d_{q-2}$ 2-dimensional grids of size $d_{q-1} \times d_q$, and each 2-dimensional grid is explored using the shadowed traversal technique as described in the 2-dimensional grids. Similarly, in the protocol ($BVD - qT$) for q-dimensional torus, the torus is partitioned into $d_1 \times \ldots \times d_{q-1}$ ring of size d_q. The exploration procedure traverses a ring and, when back to the starting point, proceeds to another ring, with a neighbouring starting point. After locating the BV, agents surround the new formed BVs sequentially and eliminate them. These strategies are simple to follow, but at the same time they are not time-efficient. Now we consider that if more than two agents are allowed to participate in the exploring

phase and we focus on decreasing the time cost in the exploring phase and the number of casualties, how to design a new strategy so that the we are able to reach the destination with acceptable cost (the increasing number of agents used in the exploring phase).

The general idea is simple, we will employ a group of agents and place them in a specific array at the beginning. Informally, in the shadowed exploration of our strategy in 2-dimensional grid(*PBVD-2G*), q-dimensional grid(*PBVD-qG*)and tori(*PBVD-qT*), the agents who are employed to explore the graph stay in that array and that "agent array" traverse that graph in the shadowed exploration. Note that after the BV is triggered, not all the agents automatically enter the elimination phase but only the agents who know the existence of BV, but in some cases, the number of agents knowing the existence of BV is not enough for surrounding and eliminating the BVs, so in the elimination phase, our strategies employ some agents who receive the clones of the original BV(thus knowing the existence of the BV) to notify some other agents to participate in the elimination phase. Also, we compare the number of agents, the time cost, the number of movement and also the casualty between each of our strategy and the sequential strategy.

4.2 Parallel BV Decontamination of Grids

4.2.1 Base Case:2-Dimensional Grid

A 2-dimensional grid (which is a mesh) of size $d_1 \times d_2$ has $n = d_1 \times d_2 (d_1 > 2, d_2 > 2)$ nodes. Without loss of generality, let $d_1 < d_2$ and let the node of M be donated by their column and row coordinate (x_1,x_2), $1 \leq x_1 \leq d_1$, $1 \leq x_2 \leq d_2$. Observe that in a mesh, we have three types of nodes: *corner* (entities with only two neighbours), *border*(entities with three neighbours), and *interior*(with four neighbours). Our strategy follows two phases: shadowed exploration and elimination. In the first phase, the network is traversed until the location of the BV is determined. That location is clear after the visit while all of

its unprotected neighbours have become BVs. Actually, in $PBVD-2G$, there are only one new formed BV. In the second phase, new formed BV is surrounded and permanently eliminated. Note that when we say the second phase starts, we actually mean that those agents knowing the existence of BV start to surround the BV, or notify some other agents, and then eliminate the BV, but not mean that all the agents enter this phase. There are two significant differences between $PBVD-2G$ and the sequential strategy: the number of agents employed in the exploration phase; the route of agents in the exploration phase. We also give the routes of agents in the elimination phase.

Shadowed Exploration Phase

As we introduce above, we should place the agents in a specific array at the beginning and then let them explore the graph. Now let us consider how to arrange them at the beginning and how to design the routes for them to explore the graph. We prefer to place the agents at the borders (or the corners) of the mesh because in this way we can reduce the casualties. For the same purpose of reducing the casualties, we prefer to arrange all the agents in a array so that when one of the exploring agent trigger the BV, the exploring agents and shadowed agents guide as many neighbours of the BV as possible. In another word, we want all the agent explore the graph in one direction but not from different direction. With these two principles, our strategy in the shadowed exploration is that given specific number of agents, we place them in one border of the mesh and if there are more agents, we place them on the row which is parallel to the border and so on. Then we design routes for them so that at any time they move to one direction to explore the graph.

Monotonicity is a principle that we should obey in the whole process, which means one exploring agent should be followed by at least one shadowed agent, so the number of agents in the exploration phase should be at least twice the number of the exploring agents. To guarantee the monotonicity and the two principles, we should employ $2a\,(a \in \mathbb{Z}^*)$ agents

in this phase and place a of them in one of the border and the others in the second line paralleling to the border. Now let's consider the number of agents we should employ. When $a = 1$, the arrangement is actually the sequential case. Since we want to explore the graph parallelly, let's start from the base case when $a = 2$. The initial arrangement would be as Fig4.1:

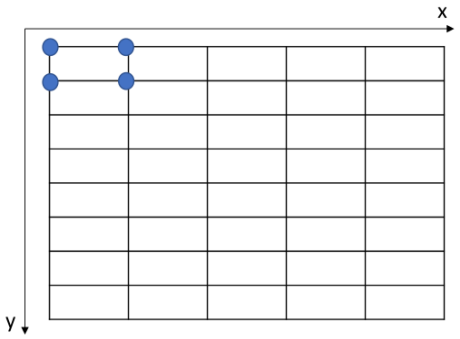


Figure 4.1: Arrangement of agents at the beginning(when $a = 2$)

Now let's consider the routes for the agents. For convenience, we assume that all the agents move at $T(x)\,(x \in \mathbb{Z}^*)$ because some time should be reserved for the coordination after the BV is triggered. More detail about the moving cycle would be discussed after we decide the number of agents and the routes for them. Let $v = (x, y)$ be the node under exploration, with $1 \leq x_i \leq d_1$, $1 \leq i \leq d_2$. Additionally, we define "Vertical Moving Mark" (VMM) for every agent: VMM can only change between 0 and 1; every time when the agent moves SOUTH, that value changes. For example, if one agent continues to move SOUTH at $T(x)\,(x \in \mathbb{Z}^*)$ and its $Vertical_D$ is originally 0, then its $Vertical_D$ changes into 1 at $T(1)$, 0 at $T(2)$ and so on. Every agent hold two VMMs in its memory and let's say VMM_1 and VMM_2. The original value of VMM_1 of the agents is 0; the original value of VMM_2 of agents residing at node $(1, y)$ and node $(2, y)$ $(1 \leq y \leq a)$ is 0 and 1 respectively.

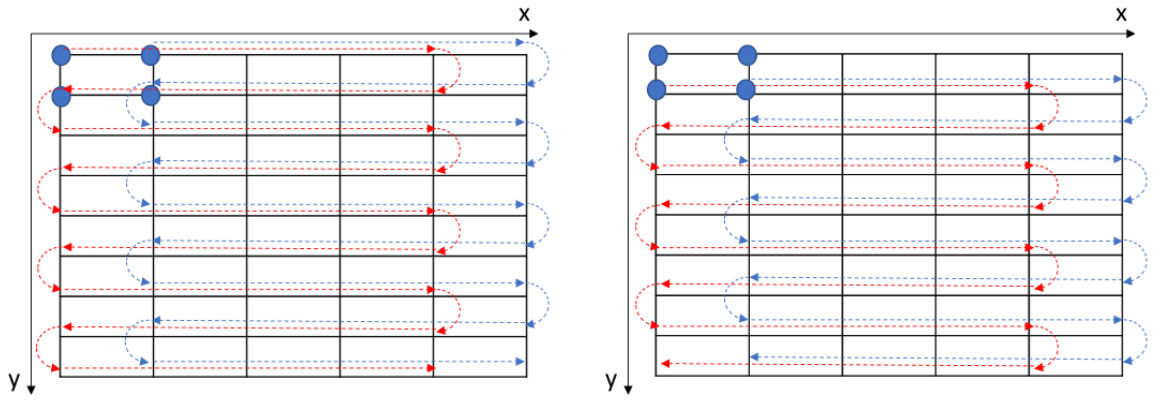
we now define the action of the $2a\,(a \in \mathbb{Z}^*)$ agents:

- Let $b = a - 1$. When all agents' VMM_1 are "0", then those agents with VMM_2 equal

to "0" move EAST when $x \neq d_1 - 1$ and move SOUTH for b steps when $x = d_1 - 1$; those agents with VMM_2 equal to "1" move EAST when $x \neq d_1$ and move SOUTH for b when $x = d_1$. When all agents' VMM_1 are "0", then those agents with VMM_2 equal to "0" move WEST when $x \neq 2$ and move SOUTH for b steps when $x = 2$; those agents with VMM_2 equal to "1" move WEST when $x \neq 1$ and move SOUTH for b steps when $x = 1$.

- A agent only move to a node which it has not explored. (Note that when residing at a node, a agent is able to know whether it has explored the neighbours of that node or not.)

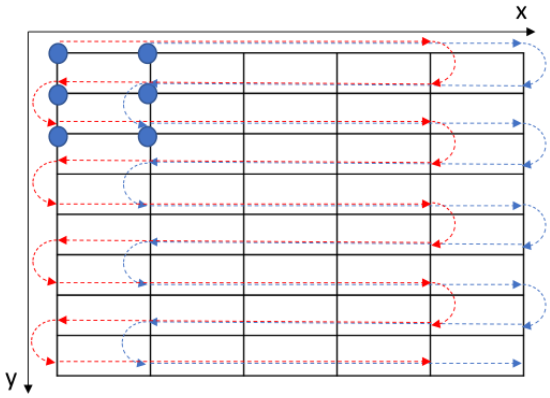
Informally, the routes of agents are snakelike routes. When $a = 2$, the routes of agents are shown as Fig4.2. In order to show the routes more clearly, we present the routes of agents in different line respectively.



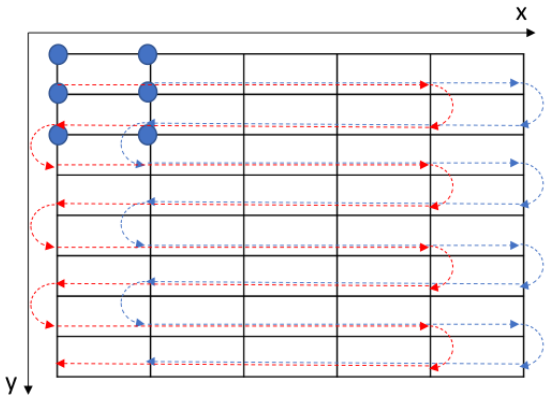
(a) Routes of agents in the first line when $a = 2$ (b) Routes of agents in the second line when $a = 2$

Figure 4.2: Routes of agents when $a = 2$

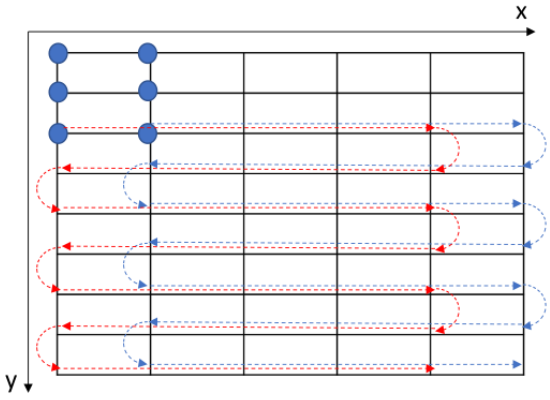
when $a = 3$, the routes of agents are shown as Fig4.3.



(a) Routes of agents in the first line when $a = 3$



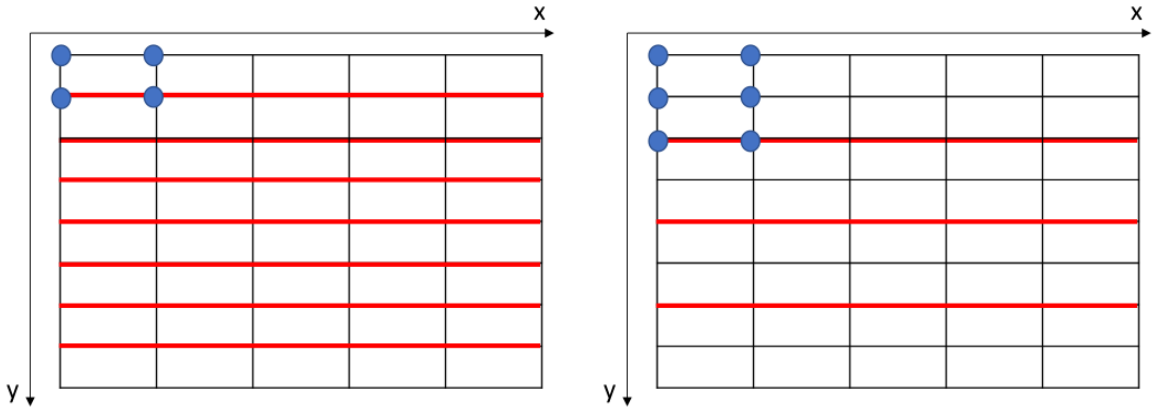
(b) Routes of agents in the second line when $a = 3$



(c) Routes of agents in the second line when $a = 3$

Figure 4.3: Routes of agents when $a = 3$

We can easily observe that some lines of the grid are traversed twice for the purpose of avoiding the explored nodes being contaminated (as shown in Fig 4.4)



(a) Lines that are double traversed when $a = 2$ (b) Lines that are double traversed when $a = 3$

Figure 4.4: Lines that are double traversed (marked with red lines)

We can see that when the grid is fixed (so d_1 is fixed), the number of double traversed lines decrease as a increases, which is easy to image. Let's denote by r the number of lines that are double traversed, then $r = \lceil \frac{d_1 - a}{a-1} \rceil$ where d_1 is the number of lines of the grid. Informally, r reflect the time that we waste in the exploration phase, and we can reduce the wasted time by employing more agents. As we can see from the equation, when $a = d_1$, then $r = 0$, which means if we employ $2d_1$ agents to explore the graph, none of the lines are traversed twice.

Now we discuss the strategy when we employ $2d_1$ agents, and this strategy can be easily modified to fit the situation where we employ less than $2d_1$ agents.

Initially, $2d_1$ agents are placed at the first two columns at T_0. More specifically, the coordinate of them are (1, x_i) and (2,x_i) where $1 \leq x_i \leq d_1$. The agents residing in the first column are in the shadowing group while the agents residing in the second column are in the exploring group. If the BV resides in a node in the first column, then all of its clones are destroyed. If the BV resides in a node in the second column, then the elimination phase begins. It is obvious that if the BV do not reside in any node in the first column, then a agent in the exploring group should be destroyed when the BV is exposed. Let's assume that the we starts at T_0. Agents residing in nodes in the second column moves EAST at

the beginning of T(2n), $n = 0, 1, \ldots, d_2 - 1$. More precisely, node (x, y) move to (x+1, y) at the beginning of T(2n), $n = 0, 1, \ldots, d_2 - 1$. Agents residing in the first column simply follow the node in the second column (see Fig.4.5). When one of the node in the former column is destroyed by a BV, the second phase starts.

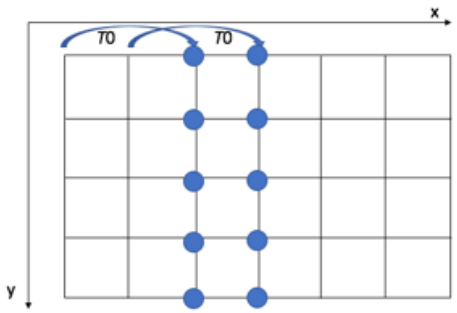


Figure 4.5: Agents move at time T_0

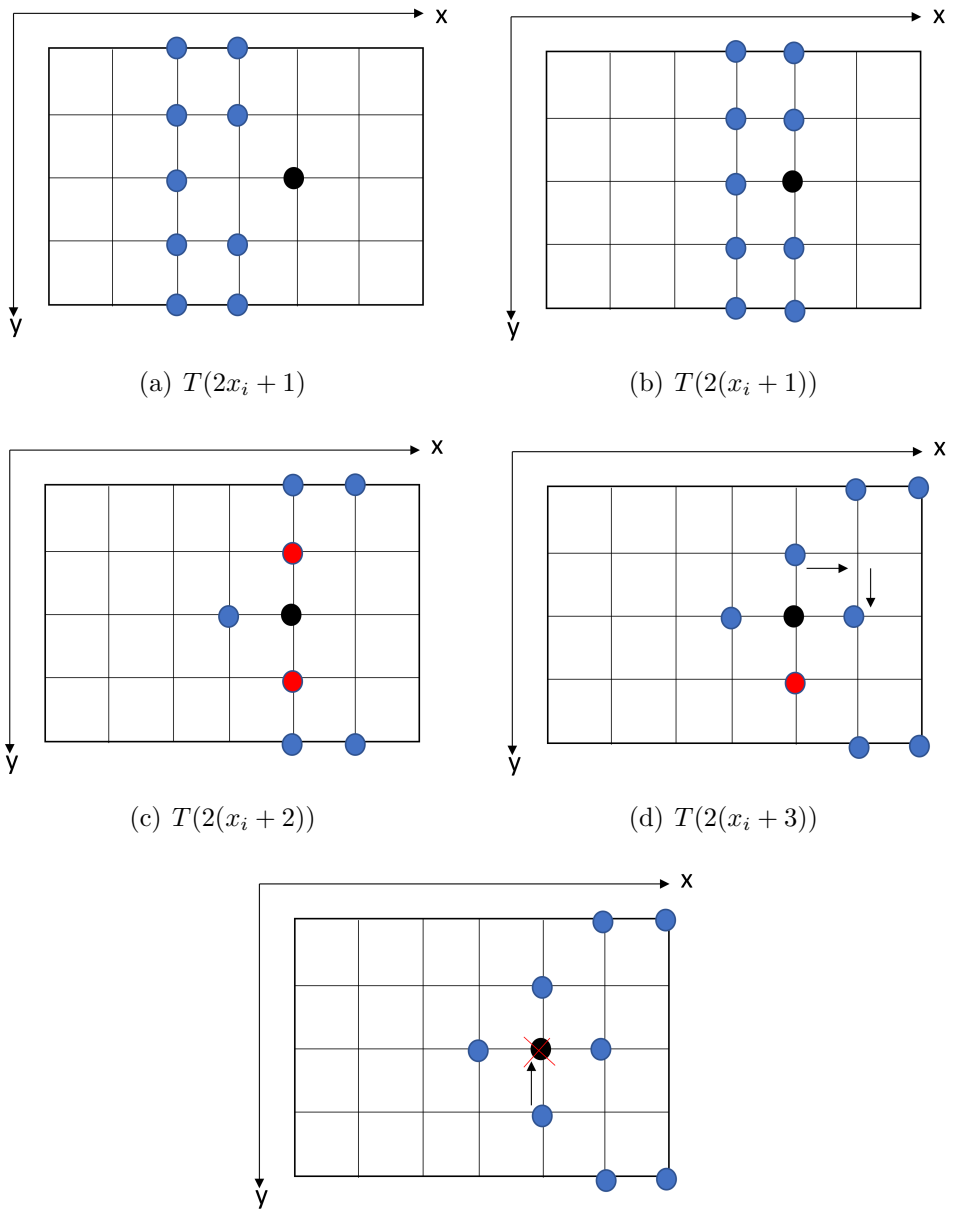
Elimination Phase

The elimination begins when one of the node in the former column is destroyed by a BV(let's say, at T($2x_i$)). No matter where is the BV, there are always three agents residing on its north(if the BV is not in the first line), west and south (if the BV is not in the last line), so only one BV clone survives. In another word, only one node becomes BV node after the BV being triggered. Observe that in the parallel strategy, not all agents participate in the elimination phase automatically when the BV is explored because only those receive the clones of the original BV and those be notified by other agents can participate in the elimination phase. Since we want to make use of the agents we employ at the beginning in the elimination phase which means the situation that the number of agents is not enough for completing the elimination phase but there are still some agents keeping exploring the graph is not allowed to happen. So in some situation, agents who receive the BV clone should notify other agents to participate in the elimination. Let the node where the surviving clone resided be (x, y) and its situation can be divided into four cases. Different routes of agents in the elimination phase are based on the location of the new formed BV.

- Case 1: When $2 < x < d_1$, $1 < y < d_2 - 1$ (a interior node becomes a new formed BV), then agents (let's say agents a, b, c) residing in node $(x-1, y+1)$, $(x-1, y-1)$ and $(x-2, y)$ receive a BV clone at T($2x_i$+1), so that they know the location of the original BV and also the new formed BV. Note that in the exploring phase, the agents move EAST at T(2n), $n = 0, 1, \ldots, d_2 - 2$. Actually, one unit of time is reserved for some agents to receive the BV clone. After they receive the BV clone, these agents move EAST for one step (for example, to node $(x, y+1)$, $(x, y-1)$ and $(x-1, y)$ and stop. Note that other agents including agents residing in node $(x-2, y+1)$ and $(x-2, y-1)$ (let's say agent d and e) at T($2x_i$) do not know the existence of the BV so they keep moving EAST. These two agents arrive at nodes $(x, y+1)$, $(x, y-1)$ at T($2(x_i+2)$) and at this time they meet agent a and b respectively. Agent a and b inform them of the location of the new form BV and the routes of agents d and e are as follow:

  route of d: $(x, y+1)(at\ T(2(x_i+2))\rightarrow(x+1, y+1)(at\ T(2(x_i+2)+1)\rightarrow(x+1, y)(at\ T(2(x_i+3))$.

  route of e: $(x, y-1)(at\ T(2(x_i+2))\rightarrow(x, y)(at\ T(2(x_i+3)+1))$. The routes of agents are showed in Fig.4.6 where a blue node indicate that there are one agent residing here; a red node indicate that there are two agents residing here; a yellow node indicate that there are three agents residing here.



(a) $T(2x_i + 1)$ (b) $T(2(x_i + 1))$

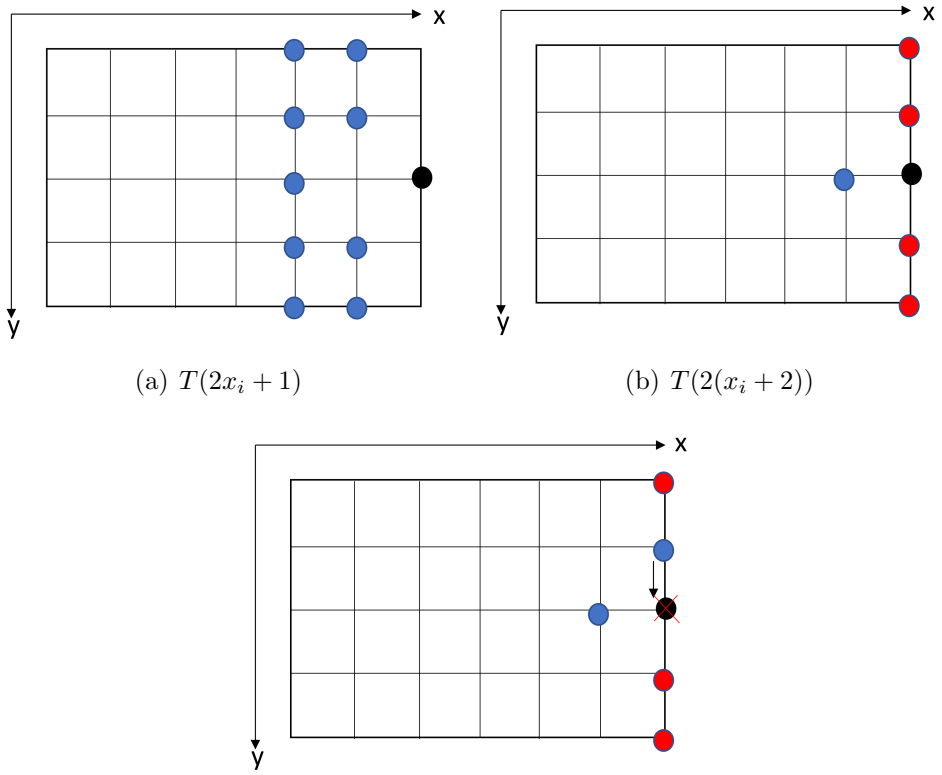
(c) $T(2(x_i + 2))$ (d) $T(2(x_i + 3))$

(e) $T(2(x_i + 3) + 1)$

Figure 4.6: Arrangement of agents in elimination phase when the new formed BV resides in a interior node

- Case 2: When $x = d_1$, $2 < y < d_2 - 1$ (a border node becomes a new formed BV), then agents (let's say agents a, b, c) residing in node $(x - 1, y + 1)$, $(x - 1, y - 1)$ and $(x - 2, y)$ receive a BV clone at T($2x_i$+1). As above, they move EAST for one step and stop. Agents residing in nodes $(x - 2, y + 1)$ and $(x - 2, y - 1)$ (let's say agents

a, b) at T($2x_i$) have no knowledge of the BV, so they keep moving and arrive at nodes $(x, y+1)$ and $(x, y-1)$ at T($2(x_i$+2)) when they are informed of the location of the new formed BV. One of agents a and b should move to the new formed BV to decontaminate it while the other one stop moving. In order to avoid conflict, we always employ the agent a to move to the new formed BV.



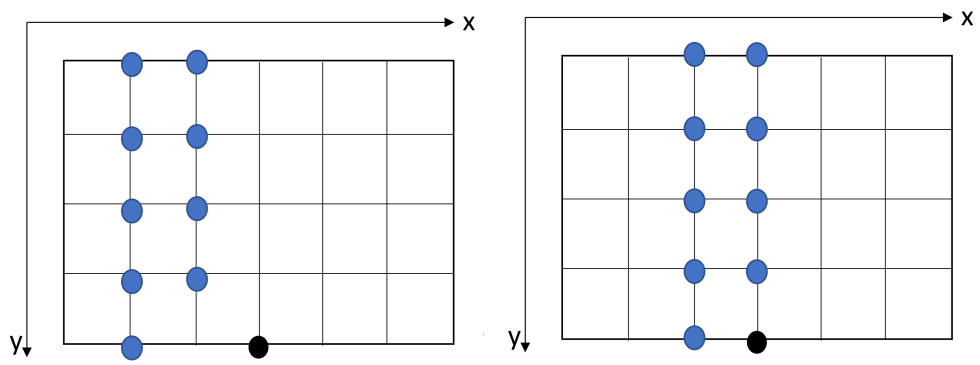
(a) $T(2x_i + 1)$ (b) $T(2(x_i + 2))$

(c) $T(2(x_i + 2) + 1)$

Figure 4.7: Arrangement of agents in elimination phase when the new formed BV resides in a border node (when $x = d_1$)

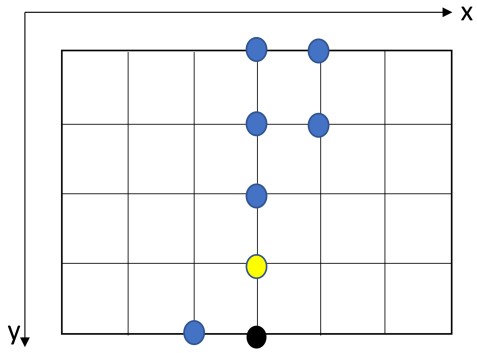
- Case 3: When $2 < x < d_1$, $y = 1$ or $y = d_2 - 1$ (a border node becomes a new formed BV). For convenience, we only discuss the situation when $y = 1$(the solution can be easily modified to fit the scenario when $y = d_2 - 1$). In this case, agents residing in nodes $(x - 1, y + 1)$ and $(x - 2, y)$ (let's say agents a,b) receive a BV clone at T($2x_i$+1). Agents a and b move EAST for one step and arrive at node $(x, y+1)$ and node $(x-1, y)$ at T($2(x_i$+1)). Since the number of agents who can be informed of the

location of the new formed BV is not enough for decontaminating the BV, we should notify one more agent to participate in the elimination phase. We employ agent a who resides in node $(x, y+1)$ at T($2x_i$+1) to notify agent residing in node $(x, y+1)$ (let's say agent c), so it move NORTH for one step and arrive at node $(x, y+2)$ at T($2(x_i+1)+1$). Then both of agents a and c move SOUTH and reach node $(x, y+1)$ at T($2(x_i+2)$) when the they meet the agent who move from node $(x-1, y+1)$ (let's say agent d) and notify it to stop moving. Then agent a move EAST and then SOUTH to reach node $(x+1, y)$ at T($2(x_i+3)$). Finally, at T($2(x_i+3)+1$), agent d move SOUTH to permanently eliminate the BV.

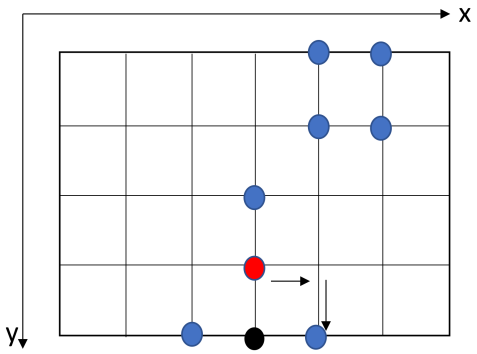


(a) $T(2x_i + 1)$

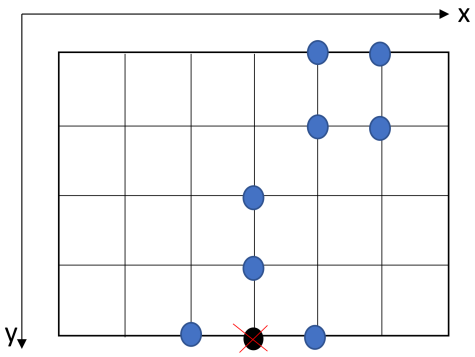
(b) $T(2(x_i + 1))$



(c) $T(2(x_i + 2))$



(d) $T(2(x_i + 3))$

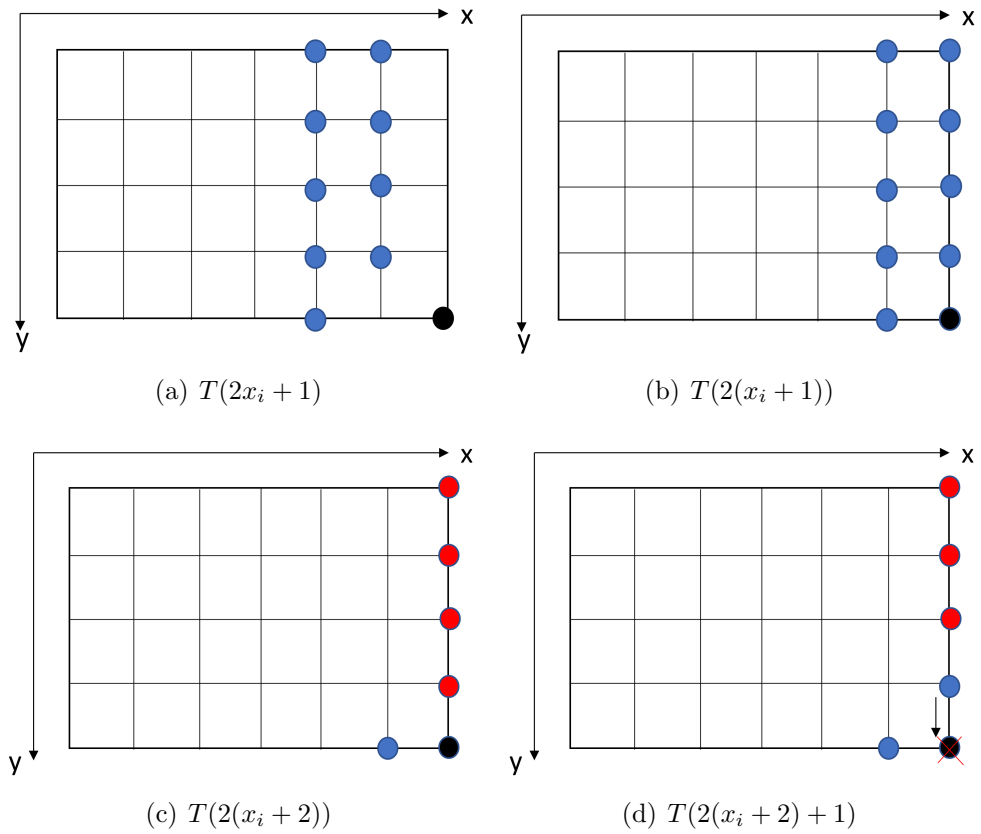


(e) $T(2(x_i + 3) + 1)$

Figure 4.8: Arrangement of agents in elimination phase when the new formed BV resides in a border node (when $y = 1$ or $y = d_2 - 1$)

- Case 4: When $x = d_1$ and $y = 1$ or $y = d_2 - 1$ (a corner node becomes a new formed BV). For convenience, we only discuss the situation when $y = 1$. In this case, agents residing in node $(x - 1, y + 1)$ and $(x - 2, y)$ (let's say agents a,b) receive a BV clone at T($2x_i$+1). Both of them keep moving for one step and arrive at nodes $(x, y + 1)$

and $(x-1, y)$ at T$(2(x_i+1))$.



(a) $T(2x_i+1)$ (b) $T(2(x_i+1))$

(c) $T(2(x_i+2))$ (d) $T(2(x_i+2)+1)$

Figure 4.9: Arrangement of agents in elimination phase when the new formed BV resides in a corner (when $x = d_1$ and $y = 1$ or $y = d_2 - 1$)

Analysis and Comparing

Theorem 1. *Mesh contamination algorithm performs a decontamination of a mesh (size of* $n = d_1 \times d_2 (d_1 > 2, d_2 > 2)$ *using* $k = 2\sqrt{n}$ *agents and at most 1 casualties.*

Proof. Let $v = (x, y)$ be the node containing the BV. When one of the agent in the exploring group moves to v, it will be destroyed and the BV will move to all neighbours of v. If $x = 1$, then the neighbours $(x, y+1)$, $(x, y-1)$ and $(x+1, y)$ are protected by agents and neighbour $(x-1, y)$ actually does not exist; if $x > 1$, then neighbours $(x, y+1)$, $(x, y-1)$ and $(x-1, y)$ are protected by agents. So when the clones of BV moves to the neighbours of v, those contain a agent will not be infected by the BV clone; this means

that the BV can safely move only to the unexplored neighbours of v, of which are at most one. In other words, after v is explored, at most one BV node are formed. According to our elimination strategy, the new formed BV node can be surrounded and destroyed using at most five agents: one to enter a BV and four to protect the neighbours. Since we have one new formed BV, the number of agents participating in the elimination phase is at most five. In addition to the agent destroyed by the original BV, the number of agent needed to complete the elimination phase is at least six. Since we employ $k = 2d_1$ ($d_1 \geq 3$) which means at the beginning we have at least six agents, so $2d_1$ agents are enough for the decontamination algorithm. □

Let us now consider the number of movements.

Theorem 2. *Parallel mesh decontamination algorithm performs a BV decontamination of a mesh of size n with at most* $2n - \sqrt{n} + O(1)$ *movements in time at most* $\sqrt{n} + 11$.

Proof. Let $v = (x, y)$ be the BV node, and let the size of the grid be $n = d_1 \times d_2$. Let us first consider the number of movements performed during the shadowed exploration. Since all the agents simply move EAST at the beginning of T(2n) ($n = 0, 1, \ldots, d_2 - 1$), the travelling distance is x for agents in the exploring group (em EA) and $x - 1$ for agents in the shadowing group(em SA). We have d_1 *EAs* and d_1 *SAs*, then we have an overall cost of at most $d_1 \times (2x - 1)$ movements for this phase. Consider now the number of movements performed for Surrounding and Elimination. In this part, we only compute the movements of the agents who participate in the Surrounding and Elimination. More specifically, we simply ignore the movements of the agents who do not know the existence of the BV in the whole process. As we discuss in the Elimination phase, when the new formed BV is located in a interior node, ten movements are needed in this phase; when the new formed BV is located in a border node (let' s say (a, b)), then eight movements are needed when $a = d_1, 2 < b < d_2 - 1$ and eleven movements are needed when $2 < a < d_1, b = 1 or b = d_2 - 1$; when the new formed BV is located in a corner node, then five movements are needed in

this phase. Hence $O(1)$ movements are performed in this phase. In total we have that the number of movements is at most $d_1 \times (2x-1) + O(1) \leq \sqrt{n} \times (2\sqrt{n}-1) + O(1)$, which is $2n - \sqrt{n} + O(1)$.

As for the time complexity. The time required for the exploration phase is equal to the number of movements of EA, which is d_1; the time required for the surrounding and elimination phase is at most eleven. So in total the parallel mesh decontamination algorithm terminate in time at most $\sqrt{n} + 11$. □

Now we make some comparison between our strategy and the sequential strategy.

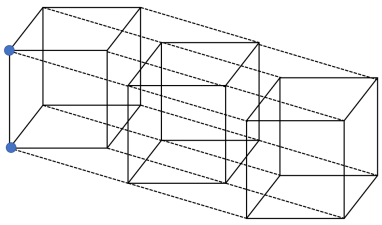
Table 4.1: Appointment Card

	The number of agent	The time cost	The number of movement	The casualties
PBVD-2G	$2\sqrt{n}$	$\sqrt{n} + 11$	$2n - \sqrt{n} + O(1)$	1
BVD-2G	7	$3n$	$9n + O(1)$	3

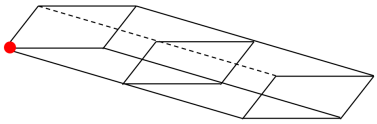
4.2.2 Multi-Dimensional Grid

Let M be a q-dimensional grid of size $d_1 \times \ldots \times d_q$ and let each node of M be denoted by its coordinates $(x_1, \ldots, x_q)$, $1 \leq x_i \leq d_q$. The algorithm, called *PBVD-qG*, follows a general strategy similar to the one described in Section 4.2.1: a safe exploration with shadowing, followed by a surrounding and elimination phase. Our general idea is that: (1) Transform the PBVD-qG problem into a BVD-qG problem (2) Use the BVD-qG strategy to solve the transformed problem. We want to solve the problem parallelly, so for convenience, if we use "q-dimensional" agent group, we use $d_1 \times \ldots \times d_p$ agents and they are organized in a $d_1 \times \ldots \times d_p$ grid. In [**?**], the multi-dimensional grid is partitioned into $d_1 \times \ldots d_(q-2)$ 2-dimensional grids of size $d_(q-1) \times d_q$. Each of these grids is explored using the BVD-2G:

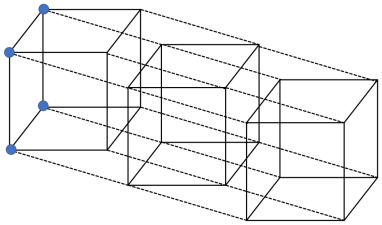
after traversing a grid in a snake-like fashion column by column, the agent returns to the starting point and from that starting point, it proceeds to another grid with a neighbouring starting point. In our strategy PBVD-qG, we use the similar exploring routes as that in BVD-G which is the snake-like route. Additionally, we use the idea of dimensionality reduction. Informally, when we solve the parallel contamination problem in q-dimensional grid using "p-dimensional" agent group, we can view the "p-dimensional" agent group as one "large agent" and view the q-dimensional grid as a "(q-p)-dimensional grid. So when we increase the number of the agents (for example, increase the "p"), then actually we decrease the dimension of the grid we want to explore. For example, if we want to solve the BVD problem in a 4-dimensional grid (say in a $d_1 \times \ d_2 \times d_3 \times d_4$ grid) using the PBVD-2G and we use d_1 agents to explore it. The problem can be viewed as using one "large" agent to explore a $d_2 \times d_3 \times d_4$ grid. If we use $d_1 \times d_2$ agents to explore it, then the problem can be viewed as using one "large" agent to explore a $d_3 \times d_4$ grid. We can choose the dimension of the agent group at the beginning and Fig4.10 shows how can we view a higher dimensional grid as a lower dimensional grid.



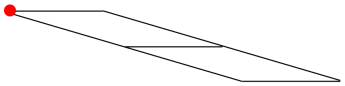
(a) The arrangement of agents on a 4-dimensional grid



(b) When we view the "one dimensional" agent as a large agent and the grid can be view as a three dimensional grid



(c) The arrangement of agents on a 4-dimensional grid



(d) When we view the "two dimensional" agent as a large agent and the grid can be view as a two dimensional grid

Figure 4.10: The idea of dimensionality reduction

After we transform the problem, then we can use the BVD-qG to solve the PBVD-qG. Let's now have a brief review of the BVD-qG. There are two agents exploring the graph: one is called exploring agent (EA) and one is called leader exploring agent (LEA). They follow the "four step casual walk" to explore the graph. Before exploring a node $v = (x_1, \ldots, x_q)(1 \leq x_i \leq d_i)$ from a node u, the shadowing agents(SA) move to the already explored neighbours of v (the coordinate of the neighbours can be precisely computed). When EA visits the BV node (and is destroyed there), the LEA and the SAs are aware of the location of the new BV nodes. (the coordinates of them can be precisely computed) So, once the v is identified, $2q$ agents surround each node $u \in N_{un}(v)$ and an additional agent enters it destroying the BV resident there and the instances that it generates.

We have map the "$d_1 \times \ldots \times d_p$"$(1 \leq p \leq q)$ agents into one "big agent". Assume that we

employ a "p dimensional" agent group (which contains $d_1 \times \ldots d_q$), then the coordinates of the agents are $(x_1, \ldots, x_p, \ldots, x_q)$, $(1 \leq x_1 \leq d_1, \ldots, 1 \leq x_p \leq d_p, x_{p+1} = 0, \ldots, x_q = 0)$. Then the coordinates of the "big agent" in the $q - p$ dimensional grid is $(x_{p+1}, \ldots, x_q)$. The first p dimensional coordinates of the agents would not change in the whole exploring phase, but from $(p+1)^{th}$ dimensional coordinates, they make the same change as the "big agent". More specifically, the $(p+1)^{th}$ of the agents make the same change as the first dimensional coordinate of the "big agent"; the $(p+2)^{th}$ of the agents make the same change as the second dimensional coordinate of the "big agent";....

Analysis and Comparing

Theorem 3. *Mesh contamination algorithm performs a decontamination of a mesh (size of $n = d_1 \times d_2 (d_1 > 2, d_2 > 2)$ using $k = 2\sqrt{n}$ agents and at most 1 casualties.*

Proof. As in the discussing in PBVD-2G, we first discuss the number of agents that should be employed in our strategy. When analyzing the PBVD-qG strategy, we use the Cost Function Measurement (CFM)

□

Chapter 5

Parallel Black Virus Decontamination in Chordal Ring

5.1 Introduction

In this chapter, we discuss a parallel strategy for BVD problem in chordal rings. A chordal ring is a circulant graph with $d_1 = 1$, i.e., it is an augmented ring, and will be denoted by $C_n(1, d_2, ..., d_k)$. More specifically, in chordal ring each node is directly connected to the nodes at distance d_i by additional links called chords. The link connecting two nodes is labeled by the distance that separates these two nodes on the ring. **Add a figure with an example of a chordal ring.**

For convenience, if we say the agents or the clones move along chord i, then they actually move along d_i. If we say the agents or the clones move i, then they actually move along chord d_x with its length equal to i.

Let us denote by d the half degree of the chordal ring (for example, for the chordal ring structure $C_n(1, 2, 4, 5)$, $d = 4$), by l the length of the longest chord of the chordal ring (for example, for the chordal ring structure $C_n(1, 2, 4, 5)$, $l = 5$). **d and l are already**

defined as k and d_k why change ? Use the same terminology throughout

In order to simplify the process, we assume that all the nodes in the network are marked with a number: the starting point is marked 0, then the second node is marked 1... **Do they have distinct Ids, or is this just done for the purpose of easier description ?**

Our goal is to minimize the time to complete the whole decontamination process and at the same time the casualties. In order to do that, we propose a parallel strategy for decontaminating the chordal ring. This is the first attempt to deal with this issue in a parallel way.

Agents are not allowed to communicate with each other unless they are in the same node so the protocol should enable agents in different nodes to move properly. That is, the route of every agent is different but they are served to explore the network; when a BV is triggered, other agents should bypass the new-formed BVs. We give simple but efficient solution to deal with the problem with acceptable cost.

The assumption on istantaneous triggering goes in the model, not here.

We would like to execute in parallel both the exploration and the elimination phase of the strategy. However, during the elimination phase a tricky situation might happen when the sites of two clones are connected. If this occurs, after these two clones are triggered, one of their clones spread to another sites and since the agents sent to destroy them die, these two sites are empty when the second round clones arrive, which make our decontamination invalid.

In order to solve this problem, we make an assumption when we talk about the parallel strategy in elimination phase in chordal rings: when a BV is triggered, its clones reach all its neighbours instantaneously.

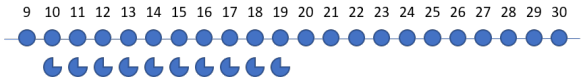
5.2 Shadowed Exploration

Initialization

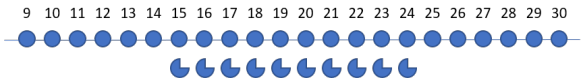
The chordal ring is a complete symmetrical structure, so we can randomly choose a node x_0 as the start node. The initial setup consists of deploying three groups of agents. Initially, we place one agent in each of the first $2d$ nodes $x_0, x_1, \ldots, x_{2d-1}$. The agents residing in nodes from x_0 to x_{d-1} form the *shadowing group*, while the ones from x_d to x_{2d-1} form the *exploring group*. If the BV is within this window of nodes, then it is easily detected. We then assume that the first $2d$ nodes do not contain the BV, and we place d additional agents at nodes $x_0, x_1, \ldots, x_{d-1}$ (*guarding group*). Only the shadowing and exploring groups move to ..... The ones in the guarding group remain dormant for now, guarding the nodes to guarantee monotonicity.

Route of the agent in exploring phase

The exploration proceeds in synchronized rounds composed each by one movement step, when selected groups of agents move to proceed with the exploration, and three steps for synchronization purposes. We call these different steps *move step* and *notification steps*. A round T_i is composed by four time units, one for the move step, and 3 for the notification steps : $T_1 = T_{move\_1}, T_{noti\_1}, T_{noti\_1}, T_{noti\_1}, T_2 = T_{move\_2}, T_{noti\_2}, T_{noti\_2}, T_{noti\_2}, \ldots$. The agents that move during a move step T_{move_i}, always do so along their longest chord d_k. That is, agents move along d_k in steps $T = 1 + 4t$ ($t \in \mathbb{N}$). An example of how agents move in chordal ring $C_n(1, 2, 4, 5)$ at T_{move_i} in the exploring phase is shown in Fig.5.1.



(a) Arrangement of agents at T_{move_i}



(b) Arrangement of agents at T_{move_i+1}

Figure 5.1: Arrangement of agents when moving

Synchronization: Three Jump Notifying Technique In the sequential strategy [] **add citation** two agents explore the graph (exploring agent and leader agent) using cautious walk. That is, the exploring agent moves to the next node in its route, and if the node is safe, it moves back to the leader agent, and then move forward to that safe node together. In this case, if the leader agent does not meet the exploring agent in next T after the exploring agent moves forward, the leader agent learns that the original BV resides in the next node so it stop moving. **doesn't it find out by receiving the clone ?** However, in our strategy, we employ $2d$ agents in the exploring phase we then have to guarantee that they all find out whether or not the BV has been found in the current round. In fact, if they are not properly informed, when one agent is destroyed by the BV, in their next step some of them (the risky agents RAs) may be destroyed by the new formed BVs. In order to avoid these potential casualties, we propose the $Three\,Jump\,Notifying\,Technique$ to properly notify the agents who otherwise would move to the new formed BVs in the next round.

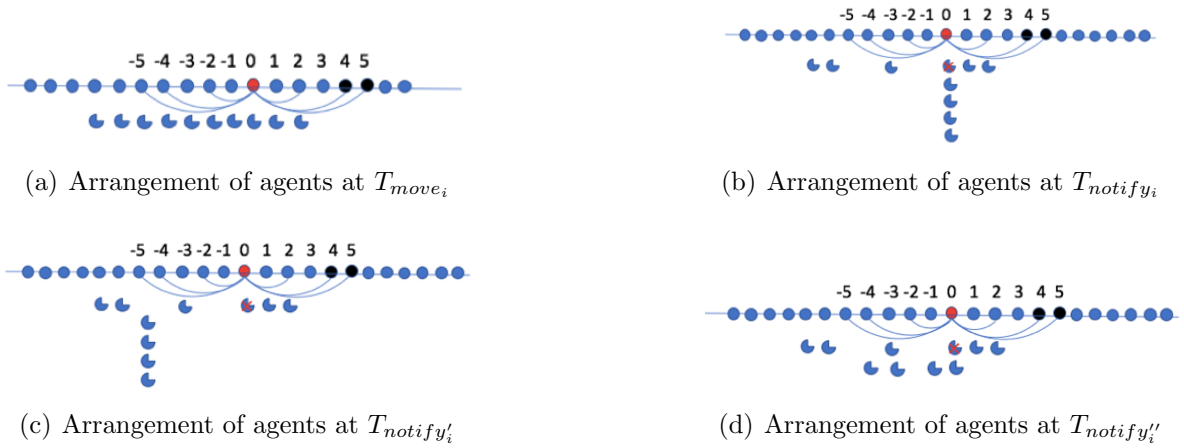
The idea is the following: when/if a node receives a clone and becomes aware of the presence of the BV, it becomes a $Notification\,Agent$ (NA). The NAs role is to make the risky agents aware of the presence of the BV in an efficient way. They will do so in parallel, each following a special route of length 3. More precisely, let the BV be at node 0 (refer to Figure 5.2), the $Notification\,Agent$ located at node $-i$ will follow the following route:

$-d_i \xrightarrow{move|d_i|} 0 \xrightarrow{move\ along\ chord\ k} -d_k \xrightarrow{move\ |d_i|} -d_k + |d_i|.$

In this case, the notifying route of the NA whose coordinate is $-d_k$ is $-d_k \rightarrow 0 \rightarrow -d_k \rightarrow 0$.

confusion between node, chord, etc. Let us always use x_i for the nodes (not i), d_i for the chords, etc..

We would make some modification of this agents route in the *Surrounding and Elimination*, but now let us assume it still follow the route above. The whole process of *Three Jump Notifying* technique in chordal ring $C_n(1, 2, 4, 5)$ is shown in Fig.5.2.



(a) Arrangement of agents at T_{move_i}

(b) Arrangement of agents at T_{notify_i}

(c) Arrangement of agents at $T_{notify'_i}$

(d) Arrangement of agents at $T_{notify''_i}$

Figure 5.2: The whole process of the *Three Jump Notifying* technique in chordal ring $C_n(1, 2, 4, 5)$

If the original BV is residing in the red node, then once an agent moves to it, the agent and the BV are destroyed but the clones of the BV spread to all its neighbours. According to our technique, agents residing in nodes -4, -3, -1, 0 are the ones to be notified; agents residing in nodes -5, -4, -2, -1 are the NAs. The routes for agents residing in nodes -5, -4, -2, -1 are $-5 \rightarrow 0 \rightarrow -5 \rightarrow 0$; $-4 \rightarrow 0 \rightarrow -5 \rightarrow -1$; $-2 \rightarrow 0 \rightarrow -5 \rightarrow -3$; $-1 \rightarrow 0 \rightarrow -5 \rightarrow -4$ respectively.

Safe Exploring with Three Jump Notifying technique

After the initialization, the exploring and shadowing agents move following the longest chord. The subsequent three time units are either used for the notification phase (if one of the agents is destroyed at time T_{move_i}), or they are spent simply by waiting before the next move. If executing the $Three\,Jump\,Notifying$ technique, the NAs move back to where they are before the notification. For example, in the example in $Three\,Jump\,Notifying$ technique, NA residing in node -1 moves back to node -4 following the reverse route in the notifying phase which is $-1 \rightarrow -5 \rightarrow 0 \rightarrow -4$.

5.3 Surrounding and Elimination

When a BV is found, the Three Jump Notifying technique guarantees that the risky agents are now aware of the presence of the BV. Other agents, however, might not have received the notification and might proceed to the next round without such knowledge; we call these agents $KeepMoving$ agents.

In this section, we introduce the process of eliminating the BVs after the original BV is triggered. For the purpose of saving the number of agents, we prefer to chase the $Keep\,Moving$ agents, but it is not necessary to complete the process especially when you care most about the execution time; In that case, we may instead carry enough agents and proceed to the $Surrounding\,and\,Elimination$ phase immediately. The number of agents that should be carried in order to successfully proceed the $Surrounding\,and\,Elimination$ will be discussed later. We now describe how to chase the $KeepMoving$ agents.

5.3.1 Notifying Moving Agents

Overview of the Notifying Moving Agents

When the $Shadowed\,Exploring$ ends, it is possible that some of the agents in the array are not informed and do not realize the existence of the BV, so they keep moving following

the routes in *Shadowed Exploring* phase but it is obvious that they would not encounter any BV. In order to reduce waste, we employ the agent who receives the clone from chord d_k (*Coordinate Agent*) to notify the other *Keep Moving* agents to move back to their position they occupied before the BV was triggered.

The Process of the Notifying Phase of the Coordination Agent

To do that we employ one of the agents as a *Coordinator Agent*(CA). **Call it coordinator agent instead of coordinate agent** The CA follows a specific path that will guarantee to meet all the *Keep Moving* within a certain amount of rounds.

The general outline of the technique is the following:

1. the CA is chosen to be the agent at distance -1 from the BV (i.e., its left neighbour)
2. once the *Three Jump Notifying* technique is terminated, the CA moves to occupy an arbitrary node within a special range $[y, z]$ (the range computation is described later).
3. the CA waits an appropriate amount of time to allow the agents that are still moving to reach this window of nodes
4. the CA now moves in synchronization with the movement of the agents and follows a specific paths. More precisely, while the moving agents proceed as usual with one move and 3 waiting steps, the CA will take its longest chord in correspondence of an agents move and 3 consecutive nodes in correspondence of the waiting steps of the moving agents. In doing so, with $O(d_k)$ moves, the CA is guaranteed to have encountered all of them.
5. when a moving agent encounters the CA, it stops and waits for a second agent that will arrive at the next round. When both agents are there, they go back to their original positions to start the surrounding phase.

—— Now try to restructure the rest describing each of the steps in separately.

We can see that the relative position of agents does not change when they keep moving along the longest chord. For example, agents residing in nodes 1, 5, 10 (noted as $A1$, $A5$, $A10$) move along the longest chord and then $A5$ moves forward for one step. Their relative position would be exactly the same as the situation where all of them remain dormant except $A5$ moves forward for one step. Now we discuss how the CA notify all the $Keep\,Moving$ agents assuming they are dormant and then make some modification to fit the scenario where the CA notifies the $Keep\,Moving$ agents. The problem we need to solve is that given a range within which the agents are in and a CA located in this range, let the CA to notify all the agents in this range. In a chord ring $C_n(1, d_2, \ldots, d_k)$, the CA sets a $Notification\,Window$ using the modular arithmetic. Let us assume the number of the node where the CA resides in is x, the number of the nodes where should be marked the $Beginning\,Flag$ and the $End\,Flag$ are y and z.

The relations between x, y, z should be as follow:

- y is the biggest number which satisfies that it is smaller than or equal to x and that $y \bmod d_k = 0$;
- z is the smallest number which satisfies that it is bigger than x and that $z \bmod d_k = d_{k-1}$.

When the agent is chosen to be the CA, it computes its $Notification\,Window$. When we mention marking a flag, it does not mean that the agent has to move to the node to do that but only needs to remember the positions of the two flags in its memory. Also, after the position of the $Notification\,Window$ is set, it remains stable. More specifically, when the CA moves, he does not set another $Notification\,Window$ using its new position. The CA moves step by step to every possible position and notifies the agents to go back

if there is one until it realizes that it just passes the $End\,Flag$. After that, he moves along the longest chord anticlockwise to the node marked a $Beginning\,Flag$ and continues moving again step by step to notify agents until it arrives its relative departing node. For example, if the CA starts to notify other from node x, then its relative departing node is $x' = x + t \times d_k$ $(t \in \mathbb{N})$.

We make some modifications to let the solution fit the real scenario where the agents move along the longest chord:

- The Beginning Flag and the End Flag move along the longest chord also to keep the relative position the same.

- When one agent $A1$ moves to a node where there is an agent $A2$ knowing the position of the original BV, $A1$ would be informed and directly moves along the longest chord to its own position.

- Let us assume that the time when the original BV is triggered is T_{move_i} $(T_{trigger})$, then the CA should remember the $T_{trigger}$ and informs the agents he encounters of it. The agent $A1$ who encounters the CA should remember the time when they encounter (T_{notify_now}) and stop moving until next T_{move} when it will meet another agent $A2$. Then $A1$ moves along d_k anticlockwise for $T_{move\_now} - T_{trigger}$ times while $A2$ moves for $T_{move\_now} - T_{trigger} + 1$ times.

- When arriving its relative departing position at T_{move_a}, the CA knows that it has finished the task and moves along d_k for $T_{move\_a} - T_{trigger} + 1$ times to its position when the original BV is triggered.

- Let us assume that the time when the BV is triggered is T_{move_i}, the CA starts to move step by step to notify other agents after T_{move_i+2}, because we want to ensure the security of the CA. If it starts to notify other agents at T_{move_i+1}, it might encounter a BV. Also, it should be ensured that the $Keep\,Moving$ agents in the exploring group

are in the $Notification\,Window$ of the CA, or the CA would never encounter them. We would talk about how to ensure this in next section.

Election of the Coordination Agent

We choose the agent who receives a BV clone from its chord d_k to be the CA, which means when an agent receives a BV clone from its longest chord, then it realizes that it is chosen as the CA. We know that, the notifying phase starts from T_{move_i+2} assuming the time when the BV is triggered is T_{move_i}, which means the notifying phase begins only after all the $Keep\,Moving$ agents move twice. At T_{move_i+2}, all the $Keep\,Moving$ agents are in a $Notification\,Window$ from nodes $d_k \times (move\_i + 1)\,to\,d_k \times (move\_i + 1) + d_k - 1$, and let us donate it by $Initial\,Notification\,Window$. The CA would directly move to any node in $Initial\,Notification\,Window$. Now we propose three kinds of routes for the CA to move to his destination:

The coordinates of the positions where the clones spread are: $x - d_k$ (which is the coordinate of the CA), $x - d_{k-1}$, $x - d_{k-2}$, $\ldots$, $x - 1$, $x + 1$, $x + d_2$, $\ldots$, $x + d_{k-1}$, $x + d_k$ supposing the coordinate of the original BV is x and we are in a chordal ring $C_n(1, d_2, \ldots, d_k)$. Now we describe three scenarios:

- Scenario 1: The last agent of the $Exploring\,Group$ is destroyed by the BV and the positions of the clones satisfy: $x - d_{k-1} + d_k = x + 1$, $x - d_{k-2} + d_k = x + d_2$, $\ldots$, $x - 1 + d_k = x + d_{k-1}$.

- Scenario 2: The last agent of the $Exploring\,Group$ is destroyed by the BV and the at least one pair of the positions of the clones does not satisfy: $x - d_{k-1} + d_k = x + 1$, $x - d_{k-2} + d_k = x + d_2$, $\ldots$, $x - 1 + d_k = x + d_{k-1}$.

- Scenario 3: One of the agents in the $Exploring\,Group$ except the last agent is destroyed by the BV.

In scenario 1, the CA needs to move for 5 steps to reach its destination while in the other two scenarios, it only needs to move for 4 steps to arrive the destination. Now we propose the route for each scenario.

- For CA in scenario 1: Let us donate by y the coordinate of the node in the $Notification\,Window$ set by the original BV which does not receive any clone and his left neighbour receives a clone (the coordinate of it is $y-1$). The CA first moves to the original BV, then to node $y-1$, finally to y. After that it only need to move along the chord d_k for twice to reach its destination.

- For CA in scenario 2: There is at least one pair of the positions of the clones does not satisfy the equations so there should be one node (assuming its coordinate is z) who receives a clone from the original BV but node $z+d_k$ is empty. The route now for the CA is first move to the BV, then to node z, and then moves along the chord d_k for twice to reach its destination.

- For CA in scenario 3: The CA here simply need to move for one step to its right neighbour and move along the chord d_k for three times to reach its destination.

In any case, the CA can reach its destination within 6 unit of time which is required for the $Keep\,Moving$ agents to move to the $Initial\,Notification\,Window$, so the CA start to chase the $Keep\,Moving$ agents as we introduce from T_{notify_i+2}.

An example of how the agents and the CA move in chordal ring $C_n(1,2,7,11)$is shown in 5.3. For our convenience, in some case, we consider the chordal ring as arranged in rows of d_k where the last node of a row is connected to the first node of the following row and the last node is connected to the first. Depending on the size of the chordal, the last row could be incomplete. So in this matrix, moving down a column corresponding to using the longest chord d_k. In the matrix, we also mark the number of nodes.

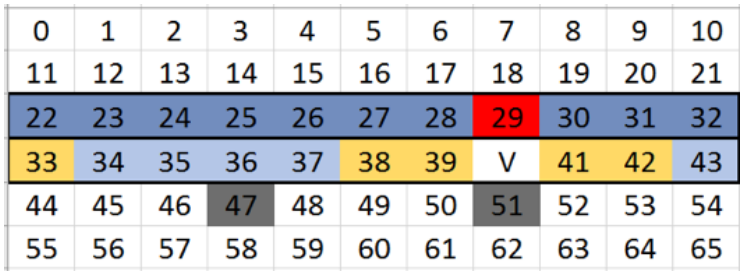


Figure 5.3: Arrangement of agent at T_{move_2} when the BV is triggered

Yellow nodes are connected to the original BV but guarded by agents while the grey nodes are the new formed BVs. The node marked V is the original BV but now is clean. Agent residing in node 29 receives clone from chord d_k so it knows it is the CA During the notifying time, agents residing in nodes 33, 38, 39 notify agents residing in nodes 36, 31, 30 respectively following the $Three\,Jump\,Notifying\,Technique$ while the CA moves to 28, 39, 50 and finally 61 following the route in scenario 3.

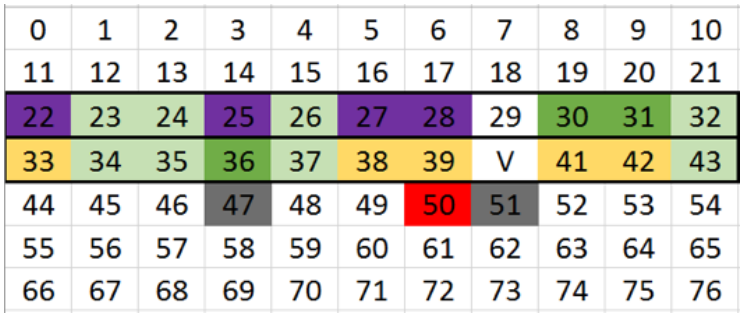


Figure 5.4: Agents roles after $Three\,Jump\,Notifying\,Technique$ and choosing CA. (for convenience, we donate the CA by a red spot, more specifically, node 50 is where CA resides)

Agents in purple nodes would be notified at T_{move_3} and move back. Agents in light green nodes are the $Keep\,Moving$ agents while agent in dark green nodes are informed to stop in $Three\,Jump\,Notifying\,Technique$. In the meantime, the CA moves to node 28, 39, 50, and finally 61. It is obvious that the CA can reach its destination before T_{move_4}, so it waits until T_{notify_4} to start its notifying phase.

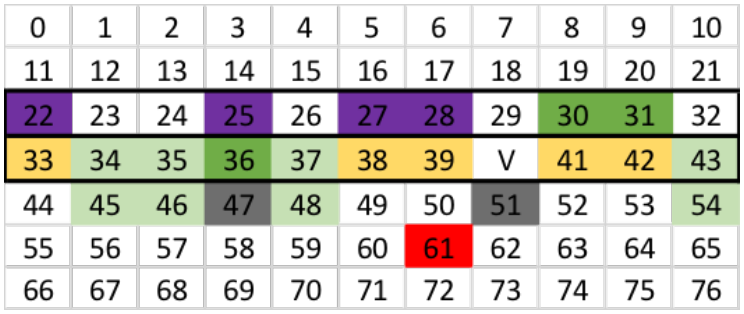


Figure 5.5: Arrangement of agent at T_{move_3}. The CA has arrived its destination)

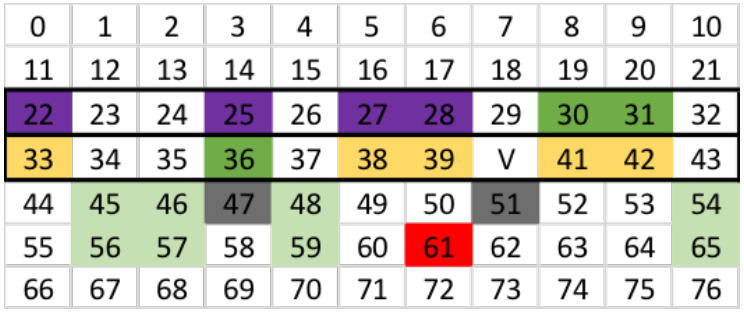


Figure 5.6: Arrangement of agents at T_{move_4}. The CA starts its notice phase

In notifying phase, CA starts to notify other $Keep\,Moving$ agents. First, it computes the $Notification\,Window$ which is from node 55 to node 65. Note that the $Notification\,Window$ would move along chord d_k at every T_{move}. It moves to node 62 at T_{notify_4}, node 63 at $T_{notify\_4'}$, node 64 at $T_{notify\_4''}$ and to node 75 at T_{move_5}.

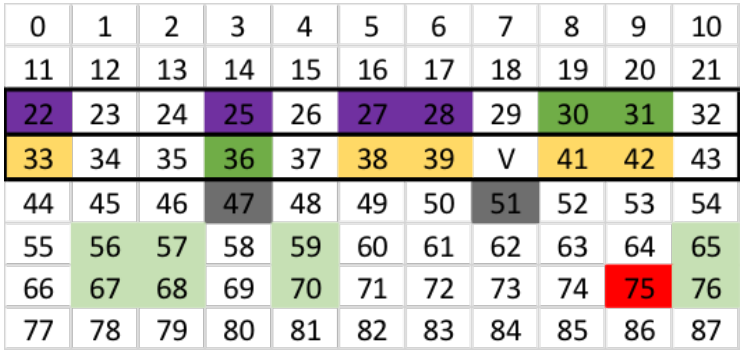


Figure 5.7: Arrangement of agents at T_{move_5}.

Again, in the notifying phase, CA moves to node 76 at T_{notify_5}. We can see that it encounters agent residing in node 76, so CA informed it the $T_{trigger}$ which is T_{move_2}. Agent

residing in node 76 should remember T_{notify_now} which is T_{notify_5} and wait until next T_{move} to inform agent (*Following Agent*) who resides in node 65 now but would move to node 76 next T_{move}. After encountering its *Following Agent*, it informs it to move back along chord d_k for $T_{move\_now} - T_{trigger} + 1$ times which is $T_{move\_5} - T_{move\_2} + 1$ times while itself moves for T_{move_5}- T_{move_2} times. At $T_{T_{notify\_5'}}$ when the CA arrives at node 77, it knows that it just pass its *Ending Flag* so it moves along the longest chord anticlockwise to its *Beginning Flag* at $T_{notify\_5''}$.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87

Figure 5.8: Arrangement of agents at $T_{move_5''}$.

0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	V	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98
99	100	101	102	103	104	105	106	107	108	109

Figure 5.9: Arrangement of agents at $T_{move_7''}$.

We could know that the CA moves back to its relative original position at $T_{notify\_7''}$. It would wait until next T_{move} to inform its *Following Agent* of $T_{trigger}$ and moves back with it.

5.3.2 Overview of the Elimination

After all the agents move back to where they are when the BV is triggered, we start the *Surrounding and Elimination*. We can destroy the BVs sequentially which is simple to execute but might in some case cost more time. In this way, because all the agents are aware of the positions of the new-formed BVs, so every time at most $d-1$ agents are sent to surround one of the BV and one agent is sent to destroy the BV and move on to decontaminate the second BV in a same way. Actually, this elimination strategy is the same as that in [**?**]. Or if we care most the execution time , we can destroy the BVs at one time. First we need to guard all the neighbouring nodes of the new formed BVs. In order to avoid collision and efficiently leverage the agents, we allocate different Destination Tables to all the agents in the array to inform them where should they should move in different situations (e.g., when the first agent in the exploring team is destroyed, then every agent except the first agent have a distinct destination, when the second agent is destroyed, then every agent except that agent destroyed have a distinct destination. More specifically, for a Chord Ring with half degree d, every agent in the array carries a $Destination\,Table$ with $d-1$ destinations. If we need more agents, then we will give their $Destination\,Table$ to the last agent in the shadowing group, when the elimination begins, it clones enough number of agents and give the $Destination\,Table$ to them. Before moving to its destination, the agent computes the shortest route from its own position to its destination using Dijkstra Algorithm. There are two kinds of agent in the Elimination phase: surrounding agents who are responsible for guarding the neighbouring nodes of the BVs and destroying agents who move to the BVs after all the neighbouring nodes are guarded. We want the BVs to be destroyed at one time, so it is important that the destroying agents move to the BVs at the same time and only after all the neighboring nodes are guarded by agents. In fact, if the destroying agents know the longest time $t_{longest}$ to move to the destination taken by all the agents (including the destroying agents and the surrounding agents), then they move to the last node prior to the destination and wait until $t_{longest}$ to move to the BVs

together. So in the *Destination Tables* for the destroying agents, we also add an item which is the $t_{longest}$. Now we introduce how to compute the shortest routes and how we design the *Destination Tables*. Note that we design *Destination Tables* for all the agents and allocate them to the agents before the exploring phase begins.

5.3.3 Destination Table and Elimination

Supposing there are some BVs and agents in the chordal ring, it is obvious that the BV nodes are in the clockwise side of the agents. In order to use Dijkstra, first we need to map the chordal ring with BVs into a graph. We include nodes from the node containing the first agent to the node which is d_k away from the last BV node, then delete the chords from the BV nodes to build the graph where we run Dijkstra Algorithm. Here is an example how we built the graph for running Dijkstra Algorithm. Below we show the situation when the third agent in the exploring group is destroyed by the BV (see 5.10). Only the chords of the original BV node are shown.

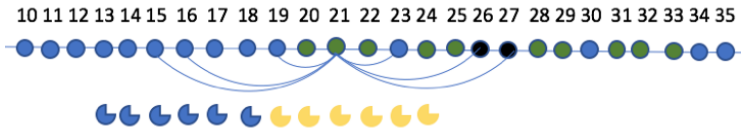


Figure 5.10: Situation when the third agent in the exploring group is destroyed

The black node is the BV node while the green nodes need to be guarded. So in this case, we need 12 agents (10 surrounding agents and 2 destroying agents). We add nodes from 13 to 33 with their chords within this area and delete chords connected with the BV nodes to get the graph where we use Dijkstra Algorithm. Below is the graph we build. (see 5.11) For convenience, we show the all the nodes we included and the chords we delete.

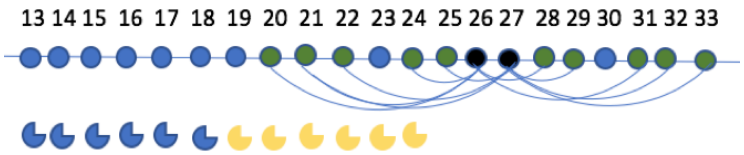


Figure 5.11: The graph we build for Dijkstra Algorithm

Using the graph and Dijkstra Algorithm, we compute the routes from every agent to every node. Then we use enumeration to choose an allocation of every agents's destination satisfying:

- 1) the maximum length of the route should be minimum.
- 2) after the allocation, in every needed position there should be exactly one agent.

After we get the optimal allocation, we record every agent?s distinct destination and the position of the third agent in the exploring group in their $Destination\,Table$. Also, we add the length of the longest chord to every destroying agent. More specifically, here we talk about the situation when the third agent in the exploring group is destroyed. For every agent, the information we compute would be in the third part of its $Destination\,Table$ recording the position of the third exploring agent (for example, it connects this agent through chord x), the destination it should move if the third exploring is destroyed. For a destroying agent, there should be another item recording the length of the longest route in this part. Above we introduce how to design one part of $Destination\,Table$ of an agent, for every agent in chordal ring, it should hold a $Destination\,Table$ of $d-1$ parts, and every part contains 2 items (for surrounding agent) or 3 items (for destroying agent). After the one of the agent is destroyed, the agent can check their $Destination\,Table$ to get the information of their destination. Then using Dijkstra Algorithm they can compute the shortest route separately and starts to move.

5.4 Analysis and Comparing

5.4.1 Theorem and Proof

Theorem 4. *If the structure of a chordal ring is fixed, no mater where is the BV, the number of Keep Moving agent is a constant.*

Proof. Define: Going column, the column in which the agents keep going when an explorer encounters a virus.

Define: Stop column, the column in which the agents stop moving when an explorer encounters a virus. We assume that the structure of the ring is $C_n(1, d_2, d_3, \ldots, d_k)$, and there are d columns in the matrix which are $M = c_0, c_1, \ldots, c_{d-1}$

We assume the number of node where the original BV is V(P_V), then it should in the column $c_{v \bmod d}$.

A column is a stop column if and only if $\exists i, j \in [1, d]$, so that $(V + r_i) \bmod d = S$ or $(V - r_i) \bmod d = S$.

If the explorer encounters the virus at the position P_{V+a} instead of P_V. Since $(V + r_i + a) \bmod d = (S + a) \bmod d$ or $(V - r_i + a) \bmod d = (S + a) \bmod d$, the column $c_{(S+a) \bmod d}$ should be a stop column.

In another word, if c_S is a stop column when virus position is P_V, then $c_{(S+a) \bmod d}$ is a stop column when the virus position is P_{V+a}.

We assume we have two different stop columns c_x and c_y , when the black virus's position is P_V. So when the black virus's position is P_{V+a}, the columns are $c_{(x+a) \bmod d}$ and $c_{(y+a) \bmod d}$. It is obviously that $\forall a$, if $x \neq y$, then $(x + a) \bmod d \neq (y + a) \bmod d$. That means we also have two different stop columns. So that the number of stop column does not decrease, which means that given a fixed chordal ring C and the number of agent keeping moving when the BV is in V (V can be any position), then the number of agent keeping moving when the BV is in other position does not decrease.

Let us assume that the number of agents keeping moving in different case when the longest chord remains the same is different and donates the minimum number of going columns among them by $N_{minimum}$ while the maximum number of going columns among them by $N_{maximum}$, then according to our conclusion $N_{minimum} \geq N_{maximum}$, which means that $N_{minimum} = N_{maximum}$. In another word, the number of agents keeping moving is a constant when the structure of the chordal ring is fixed. □

The second theorem is reserved for proving the number of agents needed in different strategies in elimination

5.4.2 Analysis and Comparing

Time cost analysis and comparing We only consider the situation when n (the number of nodes of the chordal ring) is much larger than d_k, and since the time cost in the elimination phase is $O(1)$. Finally, we compute the TWT of both protocols to present a more fair comparison. Let us assume that the total number of moves is M, then the worst case costing the most time is when the BV is located at any nodes with number from $n - d_k + 1$ to $n - 1$ and the first agent (anticlockwise) is in the node with the number $n - d_k$. In this case, it cost $M = \left\lfloor \frac{n}{d_k} - 1 \right\rfloor$ moves and $4M$ units of time ($4M = 4 \times \left\lfloor \frac{n}{d_k} - 1 \right\rfloor$) to finish the exploring phase. In [**?**], she give the number of move in three case.

- 1) In double loops the upper bound of moves is $4n - 7$.
- 2) In the triple loops, she discusses two classes of chordal ring: $C_n(1, p, k)$ and $C_n(1, d_k - 1, d_k)$.In the first case, the number of moves needed is $5n - 6d_k + 22$ while in the second case, a maximum of $5n - 7d_k + 22$ moves are needed.
- 3) In the consecutive-chordal rings, a maximum of $(d_k + 2)n - 2d_k - 3$ moves are needed.

Since in the sequential strategy, agents do not need to wait so the time cost is equal to the number of moves. And it is obvious that our protocol is much faster than the sequential strategy. But since we use much more agents, so in order to gain a fair comparison, now we compute TWT of both protocol. In the exploring phase, we use $2d_k$ agents, so the TWT of our protocol is $8n - 8d_k$. In the exploring phase of the sequential strategy, it need at least 2 agent to explore and some other shadow agents to guard the explored nodes but the number of shadow depends on the structure of the chordal ring so now we ignore them. Now we compute TWT of the sequential strategy.

- 1)In double loops the upper TWT is $8n - 14$
- 2)The TWT in chordal ring $C_n(1, p, d_k)$ is $10n-6d_k+44$ and in chordal ring $C_n(1, d_k-1, d_k)$ is $10n - 14d_k + 44$.
- 3)The TWT in consecutive-chordal rings is $2(d_k + 2)n - 4d_k - 6$.

It is obvious that when $d_k \geq 2$, our protocol is faster in first case; when $d_k \leq \frac{1}{3}n + 7$,our protocol is faster in the second case (both $C_n(1, p, k)$ and $C_n(1, d_k - 1, d_k)$); when $d > 2 - \frac{1}{n+2}$, our protocol is faster in the third case.

Calamity Analysis Casualty is the number of agents destroyed by the BV. In chordal ring $C_n(1, d_2, \ldots, d_k)$, the worst case is that the first agent in the exploring group is destroyed by a BV and the clones of it spread to all its neighbouring nodes. The casualties in this case are $d_k + 1$ because another d_k nodes are guarded by agents while in sequential case, the casualties are $2d_k$. So in terms of casualty, our protocol is better than the sequential strategy.

Following part are reserved for analysis for different strategy in elimination: including proposing function to calculate the exact cost of different strategies...